gnnjliwws

August 21, 2024

1 Naive Bayes Practive - Practice

Aug 20, 2024

1.1 1. Bài toán

Phân loại văn bản sử dụng Naive Bayes

Mục tiêu:

- Xây dựng được mô hình Naive Bayes sử dụng thư viện sklearn.
- Ứng dụng và hiểu cách áp dụng mô hình Naive Bayes vào giải quyết bài toán thực tế (*ví dụ: phân loại văn bản*).
- Sử dụng độ đo Accuracy để đánh giá chất lượng mô hình.

Vấn đề: * Có một tập các văn bản dạng text không có nhãn, làm sao để biết văn bản này thuộc về thể loại nào, pháp luật, đời sống, văn học, thể thao,...

Dữ liệu: * Tập các văn bản và nhãn tương ứng của từng văn bản trong một khoảng thời gian. * Tập các nhãn - 10 nhãn văn bản:

> Giải trí, Khoa học - Công nghệ, Kinh tế, Pháp luật, Sức khỏe, Thể thao, Thời sự, Tin khác, Độc giả, Đời sống - Xã hội.

Ví dụ văn bản nhãn thể thao:

> "Dân_trí Real Madrid đã dẫn trước trong cả trận đấu , nhưng họ vẫn phải chấp_nhận bị Dortmund cầm hòa 2-2 ở Bernabeu . Real Madrid chấp_nhận đứng thứ_hai ở bảng F Champions League …"

Bài toán: Phân loại

- Input: n vector mã hóa của các văn bản - ma trận $X = [x_1, x_2, ...x_n]$
- Output: nhãn y là 1 trong 10 nhãn trên

1.2 2. Import các thư viện cần thiết, cài thêm một số thư viện chưa sẵn có

[ ]:
```
# Cài đặt thư viện xử lý ngôn ngữ cho tiếng Việt!
!pip install pyvi
```

[ ]:
```
import os
import numpy as np
import matplotlib.pyplot as plt

# from sklearn.datasets import load_files
from pyvi import ViTokenizer  # Tách từ tiếng Việt

import sklearn.naive_bayes as naive_bayes
from sklearn.pipeline import Pipeline
from sklearn.datasets import load_files
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import learning_curve

%matplotlib inline
```

1.3 3. Load dữ liệu từ thư mục đã crawl từ trước

Cấu trúc thư mục như sau: - `data/news_1135/` - `Kinh tế/` - `bài báo 1.txt` - `bài báo 2.txt` - `Pháp luật/` - `bài báo 3.txt` - `bài báo 4.txt`

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

[ ]:
```
%cd /content/drive/MyDrive/Code_VinBigData_2024
```

[ ]:
```
data_train = load_files(container_path="data/data/news_1135/", encoding="utf-8")
print("10 files đầu:")
print("\n".join(data_train.filenames[:10]))
print("\n")
print("Tổng số files: {}".format(len(data_train.filenames)))
print("Danh sách nhãn và id tương ứng: ", [(idx, name) for idx, name in␣
 ↪enumerate(data_train.target_names)])
```

[ ]:
```
# #List tổng số file
# print(data_train.filenames)
# print()
```

[ ]:
```
### bài tập ###
# yêu cầu: hiển thị nội dung, và nhãn của văn bản đầu tiên trong tập train
# gợi ý: tự làm
################
# code

################
```

1.4 4. Tiền xử lý dữ liệu đưa dữ liệu từ dạng text về dạng ma trận

- Thử nghiệm để kiểm tra hoạt động chuyển hoá dữ liệu về dạng ma trận

[ ]:
```
# Load dữ liệu các stopwords
with open("data/data/vietnamese-stopwords.txt", encoding="utf8") as f:
    stopwords = f.readlines()
stopwords = [x.strip().replace(" ", "_") for x in stopwords]
print("Danh sách 10 từ dừng đầu tiên (từ không mang ý nghĩa phân loại): ",␣
 ↪stopwords[:10])
print()

# Transforming data
# Chuyển hoá dữ liệu text về dạng vector TF-IDF
#   - loại bỏ từ dừng
#   - sinh từ điển
module_count_vector = CountVectorizer(stop_words=stopwords)
model_rf_preprocess = Pipeline(
    [
        ("vect", module_count_vector),
        ("tfidf", TfidfTransformer()),
    ]
)

# Hàm thực hiện chuyển đổi dữ liệu text thành dữ liệu số dạng ma trận
# Input: Dữ liệu 2 chiều dạng numpy.array, mảng nhãn id dạng numpy.array

# Tiền xử lý với Bag of words
data_bow = module_count_vector.fit_transform(data_train.data, data_train.target)

# Tiền xử lý với TF-IDF
data_tfidf = model_rf_preprocess.fit_transform(data_train.data, data_train.
 ↪target)

print("10 từ đầu tiên trong từ điển:\n")
for i, (k, v) in enumerate(module_count_vector.vocabulary_.items()):
    print(i + 1, ": ", (k, v))
    if i + 1 >= 10:
        break
```

```
Danh sách 10 từ dừng đầu tiên (từ không mang ý nghĩa phân loại):  ['a_lô',
'a_ha', 'ai', 'ai_ai', 'ai_nấy', 'ai_đó', 'alô', 'amen', 'anh', 'anh_ấy']

10 từ đầu tiên trong từ điển:

1 :  ('dân_trí', 4599)
2 :  ('đêm', 16974)
3 :  ('bayern', 1535)
```

```
4 :  ('munich', 8900)
5 :  ('quật_ngã', 11580)
6 :  ('atletico', 1392)
7 :  ('madrid', 8484)
8 :  ('trận', 14456)
9 :  ('đấu', 17230)
10 :  ('vòng', 15732)
```

1.5 5. Chia dữ liệu làm 2 phần training và testing

- Training chiếm 80 % dữ liệu
- Testing chiếm 20 % dữ liệu

[ ]:
```
from sklearn.model_selection import train_test_split

# Chia dữ liệu thành 2 phần sử dụng hàm train_test_split
test_size = 0.2
# Bow
X_train_bow, X_test_bow, y_train_bow, y_test_bow = train_test_split(data_bow,␣
 ↪data_train.target, test_size=test_size, random_state=30)
# Tf-idf
X_train_tfidf, X_test_tfidf, y_train_tfidf, y_test_tfidf =␣
 ↪train_test_split(data_tfidf, data_train.target, test_size=test_size,␣
 ↪random_state=30)

# Hiển thị một số thông tin về dữ liệu
print("Dữ liệu training (BoW) =", X_train_bow.shape, y_train_bow.shape)
print("Dữ liệu testing (BoW) =", X_test_bow.shape, y_test_bow.shape)

print()

print("Dữ liệu training (TF-IDF) =", X_train_tfidf.shape, y_train_tfidf.shape)
print("Dữ liệu testing (TF-IDF) =", X_test_tfidf.shape, y_test_tfidf.shape)

print()

print("Danh sách nhãn và id tương ứng: ", [(idx, name) for idx, name in␣
 ↪enumerate(data_train.target_names)])
```

```
Dữ liệu training (BoW) = (609, 17787) (609,)
Dữ liệu testing (BoW) = (153, 17787) (153,)

Dữ liệu training (TF-IDF) = (609, 17787) (609,)
Dữ liệu testing (TF-IDF) = (153, 17787) (153,)

Danh sách nhãn và id tương ứng:  [(0, 'Giải trí'), (1, 'Khoa học - Công
nghệ'), (2, 'Kinh tế'), (3, 'Pháp luật'), (4, 'Sức khỏe'), (5, 'Thể
thao'), (6, 'Thời sự')]
```

[ ]:
```
X_train_bow[1].data
```

[ ]:
```
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  3,  1,  1,  1,  1,  1,  1,
        1,  1,  2,  1,  1,  1,  2,  5,  3,  1,  1,  1,  2,  1,  2,  1,  1,
        2,  1,  1,  1,  1,  2,  1,  1,  1,  1,  2,  2,  3,  1,  1,  1,  1,
        1,  2,  1,  1,  1,  1,  2,  1,  1,  1,  2,  1,  2,  2, 11,  3,  4,
        8,  3,  2,  2,  2,  3,  1,  1,  1,  1,  1,  3,  1,  1,  1,  1,  1,
        1])
```

[ ]:
```
### bài tập ###
# yêu cầu: Hiển thị ra id, tên nhãn của 5 văn bản đầu tiên trong tập train.
# gợi ý: lấy dữ liệu id từ biến y_train, mapping với thứ tự nằm trong mảng␣
 ↪data_train.target_names
###############
# code


###############
```

1.6 6. Training Naive Bayes model

Sử dụng thư viện sklearn để xây dựng 2 mô hình:

- `naive_bayes.MultinomialNB(alpha= 0.1)`: giá trị làm mịn alpha= 0.1
- `naive_bayes.GaussianNB()`

1.6.1 6.1. Multinomial Naive Bayes

- Sử dụng Bag of words

[ ]:
```
print("- Training ...")


# X_train.shape
print("- Train size = {}".format(X_train_bow.shape))
model_MNB = naive_bayes.MultinomialNB(alpha=0.1)
model_MNB.fit(X_train_bow, y_train_bow)

print("- model_MNB - train complete")
```

```
- Training …
- Train size = (609, 17787)
- model_MNB - train complete
```

1.6.2 6.2. Gaussian Naive Bayes

- Sử dụng TF-IDF

[ ]:
```
### bài tập ###
# yêu cầu: huấn luyện một mô hình Gaussian Naive Bayes tương tự như trên
# gợi ý: naive_bayes.GaussianNB(var_smoothing=1e-3)
###############
# code

print("- Training ...")


# X_train.shape
print("- Train size = {}".format(X_train_tfidf.shape))

print("- model_GNB - train complete")
###################
```

1.7 7. Testing Naive Bayes model

- Thực hiện dự đoán nhãn cho từng văn bản trong tập test
- Độ đo đánh giá: > accuracy = tổng số văn bản dự đoán đúng / tổng số văn bản có trong tập test

[ ]:
```
# Sử dụng thư viện tính accuracy_score trong sklearn
from sklearn.metrics import accuracy_score
```

[ ]:
```
print("- Testing ...")
y_pred_bow = model_MNB.predict(X_test_bow)
print("- Acc = {}".format(accuracy_score(y_test_bow, y_pred_bow)))
```

```
- Testing …
- Acc = 0.8431372549019608
```

[ ]:
```
# Test tương tự cho GNB
```

1.8 8. Thực hiện sử dụng model đã được train để infer 1 văn bản mới

- Dữ liệu mới đến ở dạng dữ liệu thô => cần tiền xử lý dữ liệu về dạng dữ_liệu_ma_trận
- Infer sử dụng hàm `model.predict(dữ_liệu_ma_trận)`

[ ]:
```
a = ViTokenizer.tokenize("Trường đại học bách khoa hà nội")
print(a)
```

```
Trường đại_học bách_khoa hà_nội
```

[ ]:
```
# tiền xử lý dữ liệu sử dụng module module_count_vector.
van_ban_moi = ViTokenizer.tokenize("Công Phượng ghi bàn cho đội tuyển Việt Nam")
# van_ban_moi = ["Công_phượng ghi_bàn cho đội_tuyển Việt_nam"]
print(van_ban_moi)
input_data_preprocessed = module_count_vector.transform([van_ban_moi])
```

```
print(input_data_preprocessed)

print()
print("Danh sách nhãn và id tương ứng: ", [(idx, name) for idx, name in␣
 ↪enumerate(data_train.target_names)])
```

```
Công Phượng ghi_bàn cho đội_tuyển Việt_Nam
  (0, 3769)     1
  (0, 5276)     1
  (0, 11077)    1
  (0, 15587)    1
  (0, 17601)    1

Danh sách nhãn và id tương ứng:  [(0, 'Giải trí'), (1, 'Khoa học - Công
nghệ'), (2, 'Kinh tế'), (3, 'Pháp luật'), (4, 'Sức khỏe'), (5, 'Thể
thao'), (6, 'Thời sự')]
```

[ ]:
```
### bài tập ###
# yêu cầu: dự đoán nhãn của 1 văn bản mới. Sử dụng mô hình Multinomial NB
# gợi ý: thực hiện code suy diễn mô hình từ tiền xử lý (bước 1) => infer (bước␣
 ↪4)
# chú ý: không training lại - ko gọi lại hàm fit
################
# code

################
```

1.9 9. Quan sát độ chính xác trên tập test của GNB khi thay đổi tham số var_smoothing

[ ]:
```
# code #####

var_smoothings = [1e-1, 1e-2, 1e-3, 1e-4, 1e-5]
accs = []

for var_smoothing in var_smoothings:
    model_GNB = naive_bayes.GaussianNB(var_smoothing=var_smoothing)
    model_GNB.fit(X_train_tfidf.toarray(), y_train_tfidf)

    # Hoàn thiện thêm phần code ở đây để ghi nhận acc tương ứng trong từng␣
 ↪trường hợp

# Minh họa tương quan bằng đồ thị
# Gợi ý: barplot, lineplot, logarithmic plot

#############
```

```
for i in range(len(accs)):
    print(var_smoothings[i], accs[i])
```